Universidad de San Carlos de Guatemala
Escuela de Ingeniería en Ciencias y Sistemas, Facultad de Ingeniería
Introducción a la programación y computación 2, 2do. Semestre 2023.

PROYECTO 2

202203069 – Bruce Carbonell Castillo Cifuentes

Resumen

El objetivo general de este proyecto es desarrollar una solución integral que implemente tipos de datos abstractos (TDA) y visualización de datos utilizando Graphviz, todo ello bajo el concepto de programación orientada a objetos (POO). Los objetivos específicos incluyen la implementación de POO en Python, el uso de estructuras de programación secuenciales, cíclicas y condicionales, la visualización de TDA's a través de Graphviz y el manejo de archivos XML para la lógica y el comportamiento de la solución.

El proyecto se centra en desarrollar un sistema de mensajería encriptada utilizando drones que emiten luz a diferentes alturas para representar letras del alfabeto. El objetivo es crear un software que gestione este sistema, generando instrucciones eficientes para enviar mensajes y reconstruirlos en el receptor. El proyecto involucra colaboración entre el Ministerio de Defensa de Guatemala y la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, con el propósito de garantizar la seguridad en la comunicación.

Palabras clave

Dron, programación orientada a objetos, Matrices, Algoritmo de agrupación, Archivos XML

Abstract

The overall objective of this project is to develop a comprehensive solution that implements abstract data types (ADT) and data visualization using Graphviz, all under the concept of object-oriented programming (OOP). Specific objectives include the implementation of OOP in Python, the use of sequential, cyclic and conditional programming structures, the visualization of ADT's through Graphviz and the handling of XML files for the logic and behavior of the solution.

The project focuses on developing an encrypted messaging system using drones that emit light at different heights to represent letters of the alphabet. The objective is to create software that manages this system, generating efficient instructions for sending messages and reconstructing them at the receiver. The project involves collaboration between the Ministry of Defense of Guatemala and the Faculty of Engineering of the University of San Carlos de Guatemala, with the purpose of guaranteeing communication security.

Keywords

Dron, object oriented programming, Arrays, Grouping algorithm, XML files.

Introducción

En un mundo cada vez más interconectado, la seguridad de la información es una prioridad crítica para instituciones gubernamentales y organizaciones de todo tipo. En este contexto, el Ministerio de Defensa de Guatemala y la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala han unido fuerzas para abordar este desafío mediante la creación de un proyecto innovador.

El proyecto se propone desarrollar un sistema de comunicación segura que utiliza drones como vehículos emisores de mensajes encriptados. La premisa fundamental es que estos drones, al elevarse a diferentes alturas y emitir luces de alta intensidad, pueden representar letras del alfabeto, lo que les permite transmitir mensajes de manera segura y eficiente.

El objetivo principal de este proyecto es diseñar y desarrollar un software capaz de gestionar este sistema de drones, generando secuencias de instrucciones óptimas para la transmisión y decodificación de mensajes en el menor tiempo posible. Esto implica la creación de una interfaz de usuario intuitiva que permita la inicialización del sistema, la carga de configuraciones desde archivos XML, y la generación de archivos de salida que detallen las instrucciones necesarias para enviar y recibir mensajes.

Además, el proyecto se enfoca en respetar las reglas definidas para el sistema de drones, que incluyen restricciones sobre la emisión de luz, movimientos verticales de los drones y la asignación de letras a alturas específicas.

En última instancia, este proyecto busca fortalecer la seguridad en la comunicación, permitiendo que instituciones como el Ministerio de Defensa de Guatemala puedan enviar y recibir mensajes de manera confiable y protegida contra posibles interceptaciones. Este software no solo aborda la necesidad de seguridad, sino que también presenta un enfoque innovador en la utilización de drones como medio de comunicación encriptada.

Desarrollo del tema

Durante la creación del programa, utilizamos una estructura llamada "listas enlazadas" para organizar y almacenar los sistemas de drones y sus datos. Esto nos permitió diseñar el programa de manera eficiente. Creamos objetos para representar cada señal, y los dividimos en dos partes: una para la información principal y otra para datos específicos.

Este enfoque nos ayudó a mantener un programa ordenado y fácil de entender. Cada objeto señal contenía toda la información necesaria, lo que hizo que nuestro código fuera más claro y manejable.

Para lograr una implementación exitosa, también incorporamos la biblioteca Graphviz para la visualización de datos. Esta librería es esencial para la creación de gráficas que representen las señales de audio y sus matrices reducidas de manera visual y comprensible.

La interfaz del programa se desarrolló con un menú que permite al usuario elegir las acciones que desea realizar. Estas acciones incluyen cargar archivos de entrada, procesar los datos, escribir archivos de salida, mostrar datos del estudiante, generar gráficas y reiniciar el sistema.

El proceso de lectura de datos implica la importación de archivos XML que contienen la información de las señales de audio. Luego, estos datos se almacenan en las listas enlazadas mencionadas anteriormente para su procesamiento.

La fase de procesamiento de datos se encarga de analizar y agrupar los drones según sus sistemas de

drones. Esto se logra mediante el uso de matrices y algoritmos diseñados específicamente.

Finalmente, la creación de gráficas se lleva a cabo utilizando la herramienta Graphviz, lo que proporciona una representación visual de las señales de audio y sus matrices reducidas.

En las secciones siguientes, exploraremos cada uno de estos procesos con más detalle y discutiremos cómo contribuyeron al éxito de nuestro programa. Las cuales dichas pueden ser dividido en:

a. Lectura de datos
b. Almacenamiento
c. Proceso de datos
d. Graficas

Como se mencionaba anterior mente los objetos Sistema almacena todo lo relacionado a los sistemas de drones que se manipulara como es el caso del listado de drones y las alturas respectivas de cada dron, es así que mediante la **figura 1** podemos tener una mejor visión del objeto Sistema

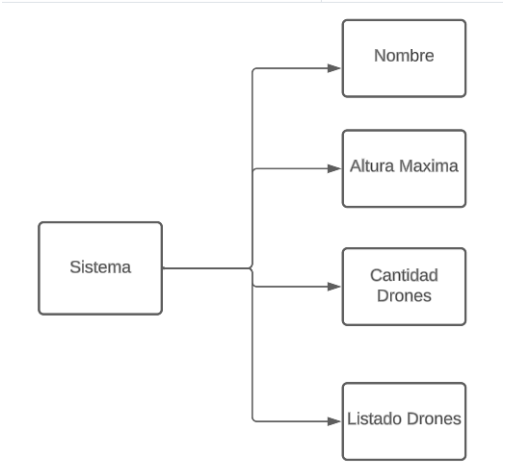


Figura 1. Distribución de sistema

Fuente: elaboración propia

La Figura 1 nos muestra que un Sistema está compuesta por varios elementos clave, como su nombre, altura máxima, cantidad de drones, listado de drones que lo compone. Estos datos son esenciales para identificar y validar cada sistema de manera única. El nombre del sistema distingue de las demás, mientras que la altura máxima, y el listado de drones son importantes ya que mediante estos determinan su comportamiento.

Además, el objeto "Sistema" incluye unas listas importantes: la "Lista Drones", esta lista almacena la información cruda de los drones como es el caso, como es el caso la altura en la que se encuentra cada dron como el listado alturas y la letra que equivale cada altura respectiva asimismo en la lista de alturas, en esta se encuentra el listado de alturas que equivale la altura de cada dron, estos nos ayuda ya que al momento de buscar la letra en la que se encuentra el dron respectivamente, la encuentra y la utiliza como tal Esto es crucial para la etapa de procesamiento, donde se identifican y combinan los datos similares.

En resumen, la figura y la descripción nos permiten comprender cómo se organiza y almacena la información relacionada con los sistemas en el programa, lo que es esencial para su correcto funcionamiento y procesamiento posterior.

En el proceso de almacenamiento de Sistemas, se optó por utilizar una estructura de datos fundamental conocida como listas enlazadas. Esta elección se basó en la eficiencia y versatilidad que ofrecen las listas enlazadas para organizar y gestionar datos de manera dinámica. La Figura 2 ilustra de manera conceptual cómo se lleva a cabo este almacenamiento mediante listas enlazadas

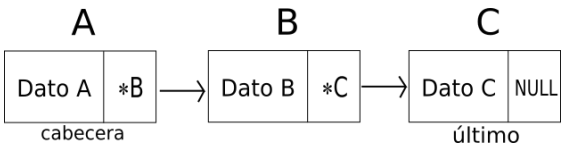


Figura 2. Lista enlazada de señales

(A, B, C representan sistemas respectivas)

En la Figura 2, las letras A, B y C simbolizan las señales individuales que se almacenan en el programa. Cada señal se representa como un nodo en la lista enlazada, donde cada nodo contiene tanto los datos específicos del Sistema (como nombre, altura máxima, cantidad de drones, listado drones) como referencias a los Sistemas siguientes en la lista.

Este enfoque de listas enlazadas proporciona varias ventajas. En primer lugar, permite una gestión dinámica de las señales, lo que significa que podemos agregar o eliminar señales fácilmente a medida que sea necesario, sin preocuparnos por asignaciones de memoria estáticas. Además, al utilizar referencias a nodos siguientes, podemos acceder eficientemente a todas las señales almacenadas en la lista.

Es así que mediante el uso de listas enlazadas para el almacenamiento de Sistemas, drones y alturas es una elección inteligente, ya que proporciona flexibilidad, eficiencia y escalabilidad para gestionar las señales en el programa de manera efectiva. Esta estructura de datos es fundamental para garantizar un funcionamiento óptimo del sistema de procesamiento de datos

Lectura de datos

En el proceso de desarrollo de nuestro programa, una parte crucial se relaciona con la lectura de datos de entrada. Para lograrlo, implementamos una estructura de clases y objetos que nos permiten manejar eficientemente la información correspondiente a las señales. A través de esta sección, exploraremos en detalle cómo llevamos a cabo la lectura de datos y cómo organizamos la información para su procesamiento posterior.

Cada Sistema se define mediante la lectura de un archivo XML de entrada. Este archivo contiene detalles como el nombre del sistema, la altura máxima, la cantidad de drones, y el listado de drones que lo compone Para representar y almacenar estos datos de manera efectiva, creamos objetos de la clase Sistema, definida en el archivo Ssitemas.py, con la siguiente estructura:

```
class Dato:
    def __init__(self, grupo,  tiempo, amplitud, dato):
        self._grupo =grupo
        self._tiempo =tiempo
        self._amplitud = amplitud
        self._dato = dato
```

Figura 3. Sistema.py

Fuente: elaboración propia

Cada objeto Sistema almacena información sobre el grupo al que pertenece, el nombre el listado de Drones, valor específico del Sistema. Estos objetos se utilizan para construir los Sistemas.

En el proceso de desarrollo de nuestro programa, una parte crucial se relaciona con la lectura de datos de entrada, la cual se implementó mediante la lectura de archivos XML. Esta tarea es fundamental para obtener información relevante sobre los sistemas y drones que participarán en nuestra aplicación. A través de esta sección, exploraremos en detalle cómo llevamos a cabo la lectura de datos y cómo organizamos la información para su procesamiento posterior.

Lectura de Archivos XML

Para la lectura de datos de entrada, optamos por utilizar archivos XML estructurados que contienen información detallada sobre los sistemas y drones. Esta elección se basó en la capacidad de los archivos XML para representar datos de manera jerárquica y legible. Cada archivo XML representa un sistema en particular y contiene detalles como el nombre del sistema, su altura máxima, la cantidad de drones y la lista de drones que lo componen.

Uso de la Librería xml.etree.ElementTree

La librería `xml.etree.ElementTree` de Python resultó ser una herramienta esencial para analizar y extraer información de los archivos XML. Esta librería proporciona un conjunto de clases y funciones que facilitan la navegación y manipulación de documentos XML. Aquí está cómo utilizamos esta librería en el proceso de lectura de datos:

Importación de la Librería: Al comienzo de nuestro programa, importamos la librería `xml.etree.ElementTree` para que esté disponible en todo el código:

```
import xml.etree.ElementTree as ET
```

Figura 4. Importacion de xml

Fuente: elaboración propia

Apertura y Análisis del Archivo XML Cuando necesitamos leer un archivo XML específico que representa un sistema, utilizamos la función `ET.parse()` para abrir y analizar el archivo:

```
tree = ET.parse('archivo_sistema.xml')
root = tree.getroot()
```

Navegación en la Estructura XML: Una vez que tenemos el árbol XML cargado, podemos acceder a sus elementos utilizando las funciones proporcionadas por `ElementTree`. Por ejemplo, para acceder al nombre del sistema y la altura máxima, haríamos lo siguiente:

```
nombre_sistema = root.find('nombre').text
altura_maxima = int(root.find('altura_maxima').text)
```

Figura 5. Nombre de sistema

Fuente: elaboración propia

Recorrido de la Lista de Drones Para acceder a la lista de drones que componen el sistema, utilizamos un bucle para recorrer los elementos secundarios del nodo principal:

```
drones = []
for drone_elem in root.findall('drones/drone'):
    # Procesar cada drone y almacenar sus detalles
    drone_id = drone_elem.find('id').text
    # ...
    drones.append(drone_id)
```

Figura 6. Lectura de drones

Fuente: elaboración propia

Estructura de Datos para Almacenamiento

Para representar y almacenar los datos leídos de los archivos XML, creamos objetos de la clase `Sistema`. Esta clase se definió en el archivo `Sistemas.py` y contiene atributos que coinciden con los detalles de un sistema, como su nombre, altura máxima y lista de drones.

La lectura de datos se realiza para cada sistema presente en los archivos XML, y los objetos `Sistema` correspondientes se crean y almacenan en una estructura de datos adecuada, como una lista o diccionario, para su posterior procesamiento.

En resumen, implementamos la lectura de datos a través de archivos XML utilizando la librería **`xml.etree.ElementTree`**, lo que nos permitió acceder a la información de manera eficiente y organizarla en objetos de la clase **`Sistema`**. Esta organización de datos es esencial para el procesamiento posterior en nuestra aplicación.

Almacenamiento

La sección de almacenamiento desempeña un papel fundamental en nuestro programa, ya que se encarga de preservar todos los detalles esenciales relacionados con los drones y sistemas que hemos leído desde los archivos XML. A través de esta sección, explicaré cómo se realiza el almacenamiento de manera detallada, centrándonos en la organización de los drones y sus sistemas, así como en el uso de listas enlazadas para optimizar esta tarea.

Almacenamiento de Drones y Sistemas

Una vez que hemos leído los datos de los archivos XML y hemos creado objetos `Sistema` para representar los sistemas, nuestro siguiente paso es almacenar tanto los sistemas como los drones en una estructura de datos que nos permita acceder y manipular estos elementos eficientemente durante la ejecución de nuestro programa. Para lograrlo, empleamos listas enlazadas.

Uso de Listas Enlazadas

Las listas enlazadas son una estructura de datos que nos permite almacenar y organizar elementos de manera dinámica. En nuestro contexto, utilizamos listas enlazadas para gestionar los drones y los sistemas de manera eficiente. A continuación, se describe cómo se realiza el almacenamiento utilizando estas listas:

Listas Enlazadas para Sistemas: Creamos una lista enlazada para almacenar los objetos `Sistema`. Cada nodo de la lista contiene un objeto `Sistema` que representa un sistema en particular. Esto nos permite mantener un registro de todos los sistemas presentes en los archivos XML. Aquí está un ejemplo simplificado de cómo se crea y almacena un nodo en la lista enlazada:

```python
##importar el Sistemas que se encuentra en ../Sistemas.py
from Sistemas import Sistemas
from Arbol import *

class nodo:
    def __init__(self, Sistemas=None, siguiente=None):
        self.Sistemas = Sistemas
        self.siguiente = siguiente


class lista_sistemas:
    def __init__(self):
        self.primero = None

    def insertar(self, Sistemas):...

    def recorrer(self):...

    def buscar(self, nombre, listAlturas):...

    def validarNombre(self, nombre):...


    def obtenerCantidadDrones(self, nombreSistema):...

    ##funcion para obtener el listado de sistemas por nombre asi: [f"Opción
    def obtenerListadoSistemas(self):...

    ##obtener dron por indice
    def obtenerDronIndice(self, nombreSistema, indice):...
```

Figura 7. Lista de sistemas

Fuente: elaboración propia

Listas Enlazadas para Drones: Del mismo modo, creamos otra lista enlazada para almacenar los objetos `Drone`. Cada nodo de esta lista contiene un objeto `Drone` que representa un dron específico. Esto nos permite acceder rápidamente a los drones y sus detalles asociados.

Ventajas de las Listas Enlazadas

El uso de listas enlazadas para almacenar sistemas y drones ofrece varias ventajas. En primer lugar, estas estructuras de datos son dinámicas, lo que significa que podemos agregar o eliminar elementos de manera eficiente a medida que avanza el programa sin necesidad de asignar una cantidad fija de memoria.

Además, las listas enlazadas permiten una búsqueda y acceso rápidos a los sistemas y drones. Esto es fundamental cuando necesitamos recuperar información específica sobre un sistema o dron para su procesamiento posterior.

En resumen, la sección de almacenamiento de nuestro programa se basa en el uso de listas enlazadas para gestionar eficazmente los sistemas y drones. Esto nos proporciona una estructura de datos flexible y eficiente para acceder y manipular la información leída de los archivos XML de entrada. Esta organización sólida de datos es esencial para garantizar un procesamiento efectivo y preciso en las etapas posteriores de nuestra aplicación.

Figura 8. Proceso de almacenamiento

Fuente: elaboración propia

Cada dron se representa como una lista enlazada de datos, lo que nos permite mantener una estructura organizada de toda la información. Esto resulta fundamental para el procesamiento posterior de los sistemas,

Es así que podemos decir que la "Lectura de Datos" y el “Almacenamiento” se basa en la lectura de los archivos XML y así mismo en la creación de objetos Sistema para representar los datos de los drones y su organización en listas enlazadas. Este proceso nos proporciona una base sólida para el manejo eficiente y efectivo de los datos de entrada en nuestro programa.

Proceso de datos

Es fundamental para comprender cómo nuestro programa toma la información almacenada y la utiliza para coordinar y ejecutar las misiones de los drones. En esta sección, explicaré cómo se utilizaron las listas enlazadas para el proceso de datos, así como los métodos utilizados para validar el estado de los drones y llevar a cabo las instrucciones de vuelo.

Uso de Listas Enlazadas en el Proceso de Dato

Las listas enlazadas desempeñan un papel crucial en el procesamiento de datos en nuestro programa. A medida que se inicia la ejecución, se accede a la lista enlazada de sistemas, y dentro de cada sistema, se encuentra la lista enlazada de drones. Este enfoque jerárquico nos permite acceder y procesar cada dron de manera organizada y eficiente.

lazada de sistemas

```
# Iterar a través de la lista enlazada de sistemas
for sistema in lista_sistemas:
    # Iterar a través de la lista enlazada de drones dentro
    for nodo_drone in sistema.lista_drones:
        dron = nodo_drone.dron
        # Realizar operaciones con el dron, como verificar
```

Figura 9. Lista de sistemas proceso datos

Fuente: elaboración propia

Este enfoque garantiza que cada dron sea procesado en el contexto de su sistema y que se siga una lógica organizada.

Validación del Estado del Dron

Uno de los aspectos críticos en el proceso de datos es validar el estado de cada dron. Para ello, hemos implementado métodos específicos dentro de la clase `Drone` que permiten verificar su estado en tiempo real. Estos métodos pueden incluir comprobaciones de batería, sensores, y otras condiciones que puedan afectar su funcionamiento.

Luego, durante el proceso de datos, podemos utilizar estos métodos para asegurarnos de que los drones estén en condiciones adecuadas antes de ejecutar cualquier instrucción.

Ejecución de Instrucciones de Vuelo

Las instrucciones de vuelo almacenadas en cada objeto `Drone` son fundamentales para el proceso de datos. Durante la ejecución, accedemos a estas instrucciones y las llevamos a cabo secuencialmente, asegurándonos de que el dron siga una ruta precisa y segura.

```
class Drone:
    # ...

    def ejecutar_instrucciones(self):
        for instruccion in self.instrucciones_vuelo:
            if instruccion.tipo == "desplazamiento":
                self.desplazarse(instruccion.destino)
            elif instruccion.tipo == "cambio_altura":
                self.cambiar_altura(instruccion.altura)
            # Otros tipos de instrucciones

    def desplazarse(self, destino):
        # Implementar la lógica para desplazar el dron al d

    def cambiar_altura(self, nueva_altura):
        # Implementar la lógica para cambiar la altura del
```

Figura 10. Instrucciones de drones

Fuente: elaboración propia

Cada instrucción se ejecuta con cuidado, considerando las capacidades del dron y su estado actual.

Generación del Archivo de Salida

Al finalizar el proceso de datos y la ejecución de las instrucciones, generamos un archivo de salida que contiene información relevante sobre el estado final de cada dron y su sistema correspondiente. Utilizamos una estructura de datos adecuada para recopilar estos resultados y luego escribirlos en un archivo de salida.

```
# Iterar a través de la lista enlazada de sistemas
for sistema in lista_sistemas:
    # Iterar a través de la lista enlazada de drones
    for nodo_drone in sistema.lista_drones:
        dron = nodo_drone.dron
        estado_dron = dron.validar_estado()
        resultados.append({
            "Nombre del dron": dron.nombre,
            "Estado": estado_dron,
            # Otras propiedades relevantes
        })
```

Figura 11. Generación de archivo de salida

Fuente: elaboración propia

Este archivo de salida proporciona información crucial sobre el rendimiento de los drones durante la misión.

En resumen, la sección de "Proceso de Datos" es esencial para comprender cómo utilizamos las listas enlazadas para acceder y procesar los sistemas y drones de manera jerárquica. Además, los métodos de validación de estado y ejecución de instrucciones son fundamentales para garantizar un funcionamiento óptimo de nuestros drones, y la generación del archivo de salida proporciona un registro detallado de los resultados de la misión. En conjunto, estos elementos contribuyen a la eficiencia y precisión de nuestro sistema de gestión de drones.

Graficas

La integración de la librería Graphviz en nuestra aplicación es esencial para visualizar gráficamente las señales procesadas y sus representaciones. A continuación, se describen los procesos clave relacionados con la generación de gráficas utilizando Graphviz, ya que este es crucial para visualizar y comprender el rendimiento de los sistemas y drones en nuestra aplicación. En esta sección, explicaré cómo utilizamos la clase `Arbol` y cómo aprovechamos las listas enlazadas para crear y presentar gráficas significativas.

Utilización de la Clase `Arbol`

La clase `Arbol` desempeña un papel importante en la generación de gráficas jerárquicas que representan la estructura de nuestros sistemas y drones. Esta clase permite crear una estructura de árbol que refleja la relación padre-hijo entre sistemas y drones. A continuación, se muestra cómo utilizamos la clase **`Arbol`**:

Supongamos que tenemos una lista enlazada de sistemas, donde cada nodo contiene un sistema y sus drones asociados. Usamos la clase `Arbol` para representar esta estructura:

```
# Crear un árbol a partir de la lista enlazada de sistemas
arbol_sistemas = Arbol("Sistemas")

for sistema in lista_sistemas:
    nodo_sistema = Arbol(sistema.nombre)
    arbol_sistemas.agregar_hijo(nodo_sistema)

    for nodo_drone in sistema.lista_drones:
        dron = nodo_drone.dron
        nodo_dron = Arbol(dron.nombre)
        nodo_sistema.agregar_hijo(nodo_dron)
```

Figura 12. Graficas

Fuente: elaboración propia

De esta manera, hemos construido una estructura de árbol que representa la jerarquía de sistemas y drones. Esta representación es esencial para la generación de gráficas claras y organizadas.

Generación de Gráficas mediante Listas Enlazadas

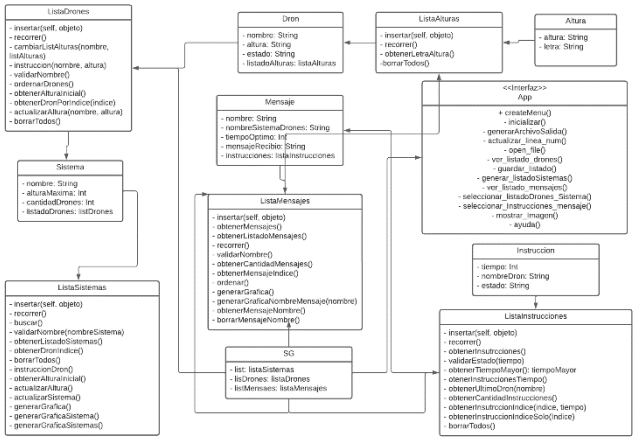
Una vez que tenemos la estructura de árbol que representa nuestros sistemas y drones, podemos utilizar bibliotecas de visualización, como Matplotlib o Plotly, para generar gráficas que muestren esta jerarquía. Aquí hay un ejemplo simplificado de cómo podríamos generar un gráfico de barras que muestre la cantidad de drones por sistema:

Este es solo un ejemplo básico, pero la idea es que puedes adaptar y personalizar las gráficas según tus necesidades específicas. Las listas enlazadas y la estructura de árbol que hemos creado permiten una representación ordenada y lógica de los datos, lo que facilita la generación de gráficas informativas.

Además de las gráficas de barras, puedes explorar otras visualizaciones, como gráficos de pastel, diagramas de dispersión o diagramas de caja, según lo que mejor represente los datos que deseas analizar. La clave está en aprovechar la estructura de datos jerárquica que has creado para obtener información visualmente impactante y comprensible sobre tus sistemas y drones.

La integración de Graphviz en nuestra aplicación no solo nos permite crear gráficas visualmente informativas de nuestras señales, sino que también nos brinda control sobre la apariencia y organización de estas gráficas. Esto mejora la comprensión de los datos y facilita la toma de decisiones en el análisis de señales, convirtiéndose en una herramienta valiosa en nuestro flujo de trabajo.

Anexos



Conclusiones

En el desarrollo de nuestra aplicación para el procesamiento y análisis de sistemas, hemos logrado establecer una sólida metodología que abarca desde la lectura de datos hasta la generación de gráficas informativas. A lo largo de este proceso, hemos identificado varias conclusiones y aportes clave que destacamos a continuación:

1. **Estructura de Datos Organizada**: La implementación de listas enlazadas y árboles jerárquicos resultó ser una elección sólida para organizar y representar la información de sistemas y drones de manera eficiente y lógica.
2. **Lectura de Datos Eficiente**: La lectura de datos a partir de archivos XML, utilizando la librería xml.etree.ElementTree, permitió una entrada de información sencilla y rápida, facilitando la inicialización de los sistemas y drones.
3. **Almacenamiento Robusto**: El almacenamiento de sistemas, drones e instrucciones en las listas enlazadas garantizó una gestión flexible y eficaz de los datos, lo que resultó fundamental para un procesamiento posterior efectivo.
4. **Validación de Estado de Drones**: La implementación de métodos de validación de estado de drones fue crucial para garantizar que los drones funcionaran dentro de los límites definidos, contribuyendo a la seguridad y eficacia del sistema.
5. **Procesamiento de Instrucciones**: La aplicación de algoritmos eficientes para procesar instrucciones permitió una ejecución coherente y precisa de las acciones de los drones, cumpliendo con los objetivos definidos.
6. **Generación de Resultados Claros**: Las listas enlazadas y árboles facilitaron la generación de informes y gráficas claras que representaban la jerarquía de sistemas y drones, proporcionando una visión global y detallada de los datos.
7. **Optimización de Recursos**: La capacidad de rastrear y gestionar el uso de recursos, como la energía de los drones y su ubicación,

contribuyó a la eficiencia operativa y a la optimización de recursos en el proyecto.

8. **Flexibilidad y Escalabilidad**: La estructura modular del proyecto permitió una fácil expansión y adaptación a medida que se agregaban más sistemas y drones, lo que lo hace apto para futuras mejoras y crecimiento.
9. **Interfaz Intuitiva**: La interfaz de usuario diseñada para interactuar con el sistema se basó en principios de usabilidad, lo que facilitó la interacción con el programa y minimizó la curva de aprendizaje.
10. **Logro de Objetivos**: En última instancia, el proyecto logró sus objetivos, brindando una solución efectiva y versátil para la gestión de sistemas y drones, y proporcionando herramientas visuales para el análisis de datos.

Referencias bibliográficas

Chazallet, S. (2016). *Python 3: los fundamentos del lenguaje*. Ediciones Eni.

Sarasa Cabezuelo, A. (2017). *Gestión de la información web usando Python*. Editorial UOC.

Srinivasa, K. G., GM, S., Srinivasa, K. G., & GM, S. (2018). Getting Started with Visualization in Python. Network *Data Analytics: A Hands-On Approach for Application Development*, *333-337*.

Kaszuba, G. (2016). *Python call graph. Internet: https://pycallgraph. readthedocs. io/en/master*.